448947 Yakup Eren Arabul

434329 Enes Dere

OOP Dersi Dönem Ödevi

Bu projenin ana amacı C++ dilini kullanarak çalışma zamanında boyutu dinamik olarak değiştirilebilen bir dizi yapısı geliştirmektir. Proje içerisinde DynamicArray sınıfı oluşturulmuş ve bu sınıftan SortedArray ve UniqueArray sınıfları türetilmiştir. Geliştirme kapsamında temel OOP prensipleri kullanılmıştır.

DynamicArray

DynamicArray sınıfı içerisindeki data, size ve capacity değişkenleri private olarak atanmıştır. Bunun nedeni bu değişkenlere doğrudan kontrolsüz bir erişim olmamasını istediğimizden dolayıdır. Bu verilere ulaşmak ve değiştirmek adına get() ve set() methodları kullanılmıştır.

DynamicArray sınıfı içerisinde dizi boyutu dolduğunda kapasitenin arttırılması adına kapasiteKontrol() methodu yazılmıştır. Bu method dizi dolduğunda dizinin boyutunu 2 katına çıkarır. Eğer kapasite yarıya düşerse ise pop() işlemi sırasında kapasite yarıya düşürülür.

Bellek Yönetimi

1)Destructor Methodu

Destructor methodu ile sınıf yok edildiğinde delete[] data komutu ile bellekte ayrılan alan belleğe geri verilir. DynamicArray içinde bu virtual olarak tanımlanmıştır. Bunun sebebi bellek sızıntısını önlemek adınadır.

2)Atama Operatörü

Mevcut nesneye başka bir nesne atandığında,eski bellek alanı temizlenir ve yeni veri Deep Copy ile aktarılır, this == &op kontrolü ile nesnenin kendine atanması durumunda gereksiz işlem yapılması engellenmiştir.

3)Copy Constructor

Bir nesne diğer bir nesnenin kopyası olarak oluşturulurken Deep Copy ile bu sağlanmıştır. Yeni bir bellek alanı ayrılarak veriler birebir kopyalanır.

Türetilmiş Sınıflar

2 adet DynamicArray sınıfından türetilmiş sınıf vardır. Bunlar SortedArray sınıfı ve UniqueArray sınıflarıdır.

1)SortedArray:

SortedArray sınıfında amacımız dizinin her zaman küçükten büyüğe sıralı kalmasını sağlamaktır.

Bunu sağlamak için DynamicArray sınıfında virtual olarak tanımladığımız push fonksiyonu override edilmiştir. Temel sınıftaki gibi elemanı sadece sona eklemek yerine önce eleman sone eklenir sonrasında ise eleman kendinden büyük olanlarla yer değiştirilerek doğru konuma getirilir.

Binary Search: Dizi her zaman sıralı olduğu için arama işleminde lineer arama yerine orta noktaya göre nereye göre ilerlemesi gerektiğini anlayan, işlemi bilgisayarlar adına daha kolay hale getiren bir algoritma kullanılmıştır.

2)UniqueArray

UniqueArray de amacımız dizinin içerisinde aynı elemanın bulunmamasını sağlamaktır.

Bu SortedArray sınıfındaki gibi push fonksiyonunun override edilmesiyle sağlanmıştır. Eklenmeden önce yardımcı bir fonksiyon olarak contains() fonksiyonu kullanılmıştır, bu fonksiyon dizinin içerisinde aynı elemanın olup olmadığını control eden bir fonksiyondur. Eğer eleman varsa ekleme işlemi iptal edilir.

Operatör Overload

1)int& operator[](int index); operatörü ile dizinin elemana ulaşmasını sağlar, & operatörü ile dizideki o elemanın kopyasına değil kendisine erişim sağlanır.

2)const int& operator[](int index) const; const koymamızın sebebi bu operatörün diziyi değiştirmesine izin vermemek adınadır.

3)DynamicArray operator+(const DynamicArray& op);
Bu operator iki diziyi uç uca ekler. Orijinal dizileri bozmaz yeni bir dizi oluşturur.
(arr3=arr1+arr2)

4) DynamicArray& operator=(const DynamicArray& op);
Bu operator bir diziyi diğerine kopyalar. (arr=arr1 gibi) Eğer arr dizisinin içi doluysa delete[] ile yer açılır, sonrasında ise Deep Copy işlemi gerçekleşir. Deep Copy sayesinde arr1 yalnızca arr dizisinin içindeki adresleri tutmak yerine kendi kendine bir dizi oluşturur. Bu sayede arr dizisinde herhangi bir değişikliğe gidildiğinde arr1 dizisi bozulmaz.

5)bool operator==(const DynamicArray& op) const;
Önce boyutlarına bakar aynı değilse direkt false döndürür ,aynı ise içerisindeki değerleri kontrol eder.

6) bool operator!=(const DynamicArray& op) const; 5. operatörün tersidir.

7) friend ostream& operator<<(ostream& os, const DynamicArray& arr);
Diziyi ekrana yazdırmaya yarayan bir operatördür. friend olmasının sebebi private olan verilere erişim kazanmasını sağlamak içindir.

Sonuç

Bu proje ile C++ dilinde bellek yönetiminin nasıl yapılması gerektiğini, bellek sızıntılarının nasıl önleneceği, override methodunun nasıl kullanılması gerektiğini, Deep Copy ile Shallow Copy arasındaki farkları, türetilmiş sınıfların nasıl kullanılması gerektiğini deneyerek öğrendik.